Combining Models to Improve Classifier Accuracy and Robustness[1]

Dean W. Abbott
Abbott Consulting
P.O. Box 22536
San Diego, CA 92192-2536 USA
Email: dean@abbott-consulting.com

Abstract
Recent years have shown an explosion in research related to the combination of predictions from individual classification or estimation models, and results have been very promising. By combining predictions, more robust and accurate models are almost guaranteed to be generated without the need for the high-degree of fine-tuning required for single-model solutions. Typically, however, the models for the combination process are drawn from the same model family, though this need not be the case.
This paper summarizes the current direction of research in combining models, and then demonstrates a process for combining models from diverse algorithm families. Results for two datasets are shown and compared with the most popular methods for combining models within algorithm families.
Key Words: Data mining, model combining, classification, boosting

1. Introduction

Many terms have been used to describe the concept of model combining in recent years. Elder and Pregibon [1] used the term *Blending* to describe "the ancient statistical adage that 'in many counselors there is safety'". Elder later called this technique, particularly applied to combining models from different classifier algorithm families, *Bundling* [2]. The same concept has been described as *Ensemble of Classifiers* by Dietterich [3], *Committee of Experts* by Steinberg [4], and *Perturb and Combine* (P&C) by Breiman [5]. The concept is actually quite simple: train several models from the same dataset, or from samples of the same dataset, and combine the output predictions, typically by voting for classification problems and averaging output values for estimation problems. The improvements in model accuracy have been so significant, Friedman *el al* [6] stated about one form of model combining (boosting) "is one of the most important recent developments in classification methodology."

There is a growing base of support in the literature for model combining providing improved model performance. Wolpert [7] used regression to combine neural network models (*Stacking*). Breiman [8] introduced *Bagging*, which combines outputs from decision tree models generated from bootstrap samples (with replacement) of a training data set. Models are combined by simple voting. Fruend and Shapire [9] introduced *Boosting*, an iterative process of weighting more heavily cases classified

[1] Presented at the 1999 International Conference on Information Fusion—Fusion99, Sunnyvale, CA, July 6-8, 1999.

incorrectly by decision tree models, and then combining *all* the models generated during the process. *ARCing* by Breiman [5] is a form of boosting that, like boosting weighs incorrectly classified cases more heavily, but instead of the Fruend and Shapire formula for weighting, weighted random samples are drawn from the training data. These are just a few of the most popular algorithms currently described in the literature, and researchers have developed many more methods as well.

While most of the combining algorithms described above were used to improve decision tree models, combining can be used more broadly. Trees often show benefits from combining because the performance of individual trees are typically worse than other data mining methods such as neural networks and polynomial networks, and because they tend to be structurally unstable. In other words, small perturbations in training data set for decision trees can result in very different model structures and splits. Nevertheless, results for any data mining algorithm that can produce significant model variations can be improved through model combining, including neural networks and polynomial networks. Regression, on the other hand, is not easily improved through combining models because it produces very stable and robust models. It is difficult through sampling of training data or model input selection to change the behavior of regression models significantly enough to provide the diversity needed for combining to improve single models.

While the reasons combining models works so well are not rigorously understood, there is ample evidence that improvements over single models are typical. Breiman [5] demonstrates bagging and arcing improving single CART models on 11 machine learning datasets in every case. Additionally, he documents that arcing, using no special data preprocessing or classifier manipulation (just read the data and create the model), often achieves the performance of handcrafted classifiers that were tailored specifically for the data.

However, it seems that producing relatively uncorrelated output predictions in the models to be combined is necessary to reduce error rates. If output predictions are highly correlated, little reduction in error is possible as the "committee of experts" have no diversity to draw from, and therefore no means to overcome erroneous predictions. Decision trees are very unstable in this regard as small perturbations in the training data set can produce large differences in the structure (and predictions) of a model. Neural networks are sensitive to data used to train the models and to the many training parameters and random number seeds that need to be specified by the analyst. Indeed, many researchers merely train neural network models changing nothing but the random seed for weight initialization to find models that have not converged prematurely in local minima. Polynomial networks have considerable structural instability, as different datasets can produce significantly different models, though many of the differences in models produce correlated results; there are many ways to achieve nearly the same solution.

A strong case can be made for combining models across algorithm families as a means of providing *uncorrelated* output estimates because the difference in basis functions used to build the model. For example, decision trees produce staircase decision boundaries via rules effecting one variable at a time. Neural networks produce smooth decision boundaries from linear basis functions and a squashing function, and polynomial networks use cubic polynomials to produce an even smoother decision boundary. Abbott [10] showed considerable differences in classifier performance class by class—information that is clear to once classifier is obscure to another. Since it is difficult to gauge *a priori* which algorithm(s) will produce the lowest error for each domain (on unseen data), combining models across algorithm families mitigates that risk by including contributions from all the families.

2. Method for Combining Models

Model combining done here expands on the bundling research done by Elder [2]. Models from six algorithm families were trained for each dataset. To determine which model to use for each algorithm family, dozens to hundreds of models were trained and only the single best was retained; only the best model from each algorithm family was represented.

2.1. Algorithms and Combining Method

Once the six models were obtained, they were combined in every unique combination possible, including all two, three-, four-, five-, and six-way combinations. Each of the combinations was achieved by a simple voting mechanism, with each algorithm model having one vote. To break ties, however, a slight weighting factor was used, with the models having the best performance during training given slightly larger weight (Table 2.1). For example, if an example in the evaluation dataset had one vote from a first-ranked model, and another from a second-ranked model, the first-ranked model would win the vote 1.28 to 1.22. The numbers themselves are arbitrary, and only need to provide a means to break ties.

Table 2.1: Model Combination Voting Weights to Break Ties

Model Rank on Training Data	Weight
First	1.28
Second	1.22
Third	1.16
Fourth	1.10
Fifth	1.05
Sixth	1.00

The six algorithms used were neural networks, decision trees, k-nearest neighbor, Gaussian mixture models, radial basis functions, and nearest cluster models. Five of the six models for each dataset were created using the PRW by Unica Technologies [11], and the sixth model (C5 decision trees) was created using Clementine by SPSS [12]. Full descriptions of the algorithms can be found in Kennedy, Lee, *et al* [13].

2.2. Datasets

The two datasets used are the glass data from the UCI machine learning data repository [14] and the satellite data used in the Statlog project [15]. Characteristics of the datasets are shown in Table 2.2:

Table 2.2: Dataset Characteristics

	Number Examples			Number Inputs/Outputs	
Dataset	Train	Test	Eval	Vars	Classes
Glass	150	0	64	9	6
Satellite	3105	1330	2000	36	6

Training data refers to the cases that were used to find model weights and parameters. Testing data was used to check the training results on independent data, and was used ultimately to select which model would be selected from those trained. Training and testing data split randomly, with 70% of the data used for training, 30% for testing. No testing data was used for the glass dataset because so few examples were available; models were trained and pruned to reduce the risk of overfitting the data.

A third, separate dataset, the evaluation dataset, was used to report all results shown in this paper. The evaluation data was not used during the model selection process, only to score the individual and combined models, so that bias would not be introduced. The glass data was split in such as way as to retain the relative class representation in both the training and evaluation datasets.

A breakdown of the number of cases per class is shown in following two tables, 2.3 and 2.4:

Table 2.3: Glass Data Class Breakdown

	Number Examples	
Class	Train	Eval
1	49	21
2	54	22
3	12	5
5	9	4
6	6	3
7	20	9

Note that there are no examples for class 4.

Table 2.4: Satellite Data Class Breakdown

	Number Examples		
Class	Train	Test	Eval
1	752	320	461
2	323	156	224
3	649	312	397
4	283	132	212
5	343	127	237
7	755	283	470

Note that there are no examples for class 6.

3. Results

Results are compiled single models for each of the six algorithms, and all possible model combinations.

Table 3.1: Number of Model Combinations

Number Models	Number Combos
1	6
2	15
3	20
4	15
5	6
6	1

The emphasis here is on minimizing classifier error without going through the process of fine-tuning the classifiers with domain knowledge to improve performance—a necessary step for real-world applications.

3.1. Glass Dataset Results

The single best models for each algorithm family is shown in Figure 3.1 below. Results are presented in terms of classification errors, so smaller numbers (shorter bars) are better. For each model, a search for the best model parameters was performed first, increasing the likelihood that the best model for each algorithm was found.

Nearest neighbor had perfect training results (by definition), and the best remaining algorithms were, in order, neural networks, decision trees, Gaussian mixture, nearest cluster, and radial basis functions, and ranged from 28.1% error to 37.5% error. Interestingly, nearest cluster and Gaussian mixture models, both using PDF measures, had the best on evaluation data.

Model combinations produced the following results shown in Figure 3.2. Not all datapoints can be seen as model combinations sometimes produce identical error scores. Two interesting trends can be seen in the figure. First, the trend is for the percent classification error to decrease as the number of models combined increases, though the very best (lowest classification error) case occurs with 3 or 4 models. The lower error rate (23.4%) occurs for the combinations in Table 3.1 below.

Amazingly, radial basis functions occur in all four of the best combination, even though it was clearly the single worst classifier. Each of the other classifiers was represented exactly twice except the Gaussian mixture which occurred once. Radial basis functions also appeared in two of the four *worst* combinations of more than 3 classifiers as well (Table 3.2), so it appears this algorithm is a wild card, and one cannot tell from the training result alone whether or not it will combine well. The worst 2-way models always include neural networks or k-nearest neighbor, and in these cases, the models were not improved compared to the single model results (34.4% for k-nearest neighbor, 31.3 for neural networks).

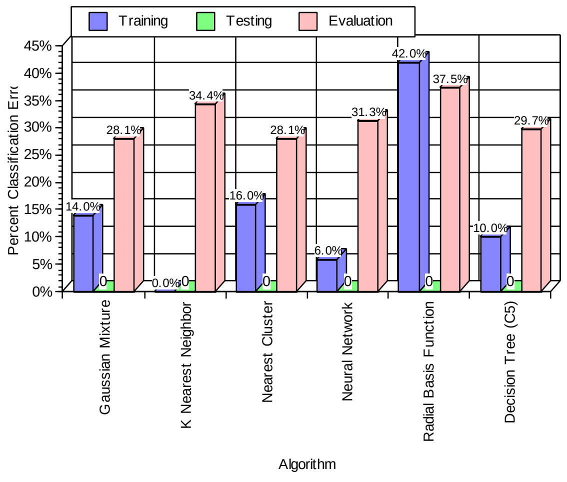

Figure 3.1: Single Model Results on Glass Data

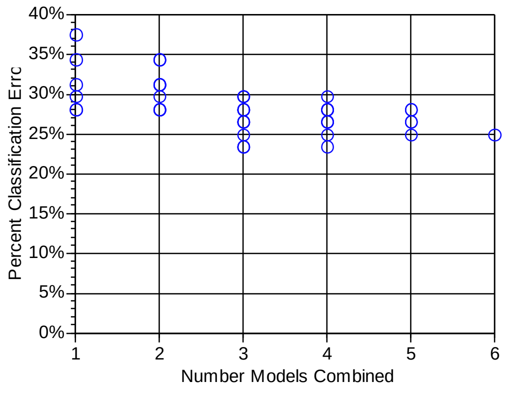

Figure 3.2: Combine Model Results on Glass Data

Table 3.1 Combinations Yielding Lowest Error Rate

k-Nearest Neighbor		Neural Networks	Radial Basis Functions
Decision Trees	Nearest Cluster		Radial Basis Functions
Decision Trees	Gaussian Mixture		Radial Basis Functions
k-Nearest Neighbor	Nearest Cluster	Neural Networks	Radial Basis Functions

Table 3.2 Greater than 3-way Combinations Yielding Highest Error Rate (29.7%)

Decision Trees	k-Nearest Neighbor	Nearest Cluster		
Decision Trees,			Radial Basis Fn.	Neural Networks
	k-Nearest Neighbor	Nearest Cluster	Radial Basis Fn.	
	k-Nearest Neighbor	Nearest Cluster		Neural Networks

When the combination model results are represented only by the error summary statistics (minimum, maximum, and average), the trends become clearer, as seen in Figure 3.3.

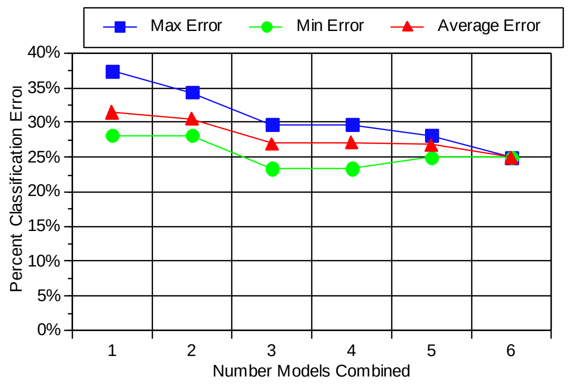

Figure 3.3: Combine Model Trend on Glass Data

The average model error never gets worse as more models are added to the combinations. Additionally, the spread between the best and the worst shrinks as the number of models combined increases: both bias and variance are reduced: the error was reduced by 4.7%, a 16.7% error reduction compared to the best Gaussian mixture model which had 28.1% error. However, the reduction found here is not as good as the reduction found by Brieman [5] using *boosting* (Figure 3.4), which brought the error down to 21.6%.

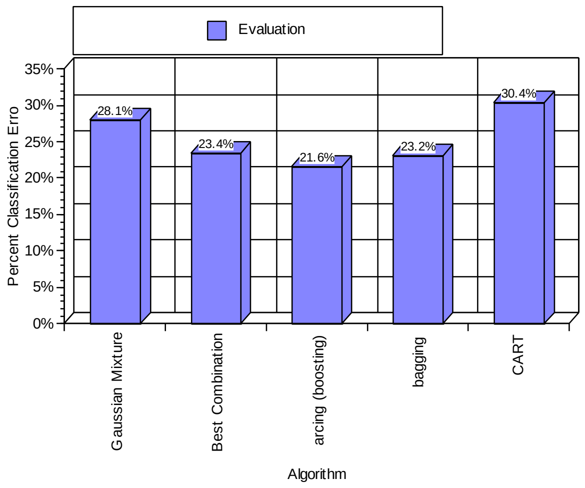

Figure 3.4: Comparison of Model Combination with Breiman Arcing.

3.2. Satellite Dataset Results

Results for the satellite data are similar to the glass data. First see in Figure 3.5 the train, test, and evaluation results for the single models. Results are more uniform than for the glass data, but radial basis functions and decision trees are the worst performers on evaluation data. The best are nearest neighbor, neural networks, and Gaussian mixture models.

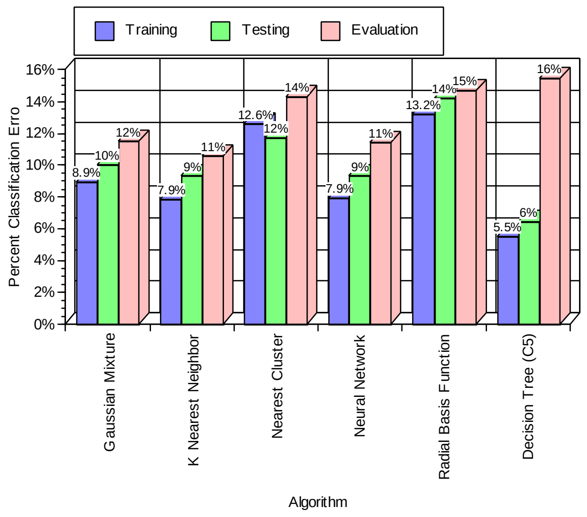

Figure 3.5: Single Model Results on Satellite Data

The trends are shown in Figure 3.6. Once again the errors and the spread between maximum and minimum errors are both reduced as the number of combined models increases, though once again the very best models occur for the 3-way combination (k-Nearest Neighbor, Neural Network, Radial Basis Function, and the same three with Decision Trees). Once again, radial basis functions are involved in the best combination, and again are also involved in the worst combination models.

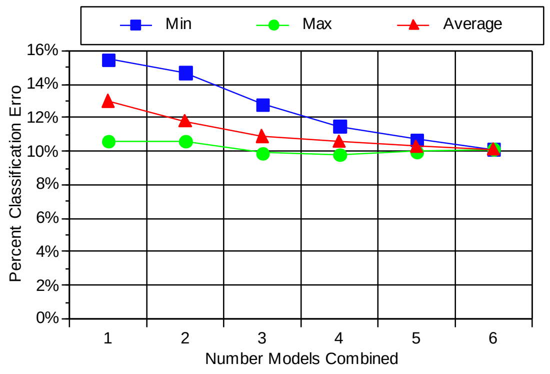

Figure 3.6: Combine Model Trend on Satellite Data

Comparing the model combination results to Breiman's results using Arcing (boosting) shows once again the boosting algorithm performing better, though the combination betters bagging by a small amount here.

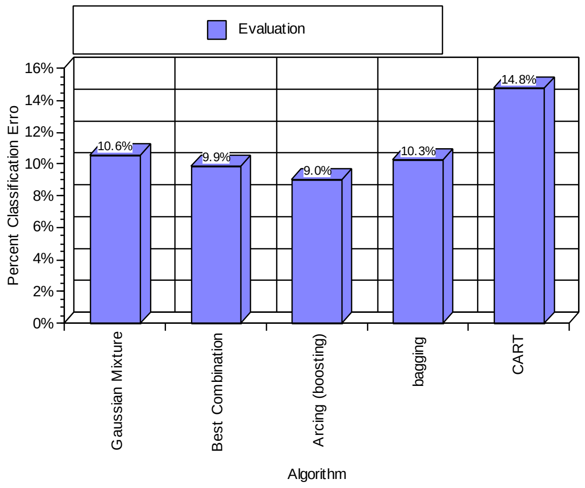

Figure 3.7: Comparison of Model Combination with Breiman Arcing.

4. Conclusions and Discussion

Clearly, combining models improves model accuracy and reduces model variance, and the more models combined (up to the number investigated in this paper), the better the result. However, determining which individual models combine best from training results only is difficult—there is no clear trend. Simply selecting the best individual models does not necessarily lead to a better combined result.

While combining models across algorithm families reduces error compared to the best single models, it does not perform as well as boosting. The advantage of boosting over simple model combining is that boosting acts directly to reduce error cases, whereas combining works indirectly. The model combining voting methods are not tuned to take into account the confidence that a classification decision is made correctly, nor do they concentrate more heavily on the difficult cases. More research is necessary to confirm these suggested explanations.

References

[1] Elder, J.F., and Pregibon, D. 1995. A Statistical Perspective on Knowledge Discovery in Databases. *Advances in Knowledge Discovery and Data Mining*. U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Editors. AAAI/MIT Press.

[2] Elder, J. F. IV, D.W. Abbott, Fusing Diverse Algorithms,. *29th Symposium on the Interface*, Houston, TX, May 14-17, 1997.

[3] Dietterich, T. 1997. Machine-Learning Research: Four Current Directions. *AI Magazine*. 18(4): 97-136.

[4] D. Steinberg, *CART Users Manual*, Salford Systems. 1997.

[5] Breiman, L. 1996. "Arcing Classifiers", *Technical Report*, July 1996.

[6] J. Friedman, T. Hastie, and R. Tibsharani, "Additive Logistic Regression: A Statistical View of Boosting", Technical Report, Stanford University, 1998.

[7] D.H. Wolpert, Stacked generalization, *Neural Networks* **5**: 241-259, 1992.

[8] L. Breiman, "Bagging predictors", *Machine Learning* **24**: 123-140, 1996.

[9] Y. Freund, and R.E. Shapire, "Experiments with a new boosting algorithm", In *Proceedings of the Thirteenth International Conference on Machine Learning*, July 1996.

[10] Abbott, D.W., "Comparison of Data Analysis and Classification Algorithms for Automatic Target Recognition", *Proc. of the 1994 IEEE International Conference on Systems, Man, and Cybernetics*, San Antonio, TX, October, 1994.

[11] Unica Technologies, Inc., http://www.unica-usa.com

[12] SPSS, Inc., http://www.spss.com/software/clementine

[13] R.L. Kennedy, Y. Lee, BV Roy, C.D. Reed, and R.P. Lippman, *Solving Data Mining Problem Through Pattern Recognition*, Prentice Hall, PTR, Upper Saddle River, NJ, 1997.

[14] http://www.ics.uci.edu/~mlearn/MLRepository.html

[15] D. Michie, D. Spiegelhalter, and C. Taylor, *Machine Learning, Neural and Statistical Classification*, Ellis Horwood, London, 1994.